Design Pattern을 이용한 CBTC 차상시스템의 모델링

Pattern-Based Modeling of CBTC Onboard System

임재식* 한재문* 양찬석* 김형훈** 조용기*

Lim, Jae Shik Han, Jae-Mun Yang, Chan-Seok Kim, Hyoung-Hoon Cho, Yong-Gee

ABSTRACT

A design pattern is a general reusable solution to a commonly occurring problem in software design. A design pattern is not a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations. Moreover, the name of pattern itself also can be a kind of common language among developers and patterns can be easily imported to various applications demanding similar requirements. In this paper, we present models of CBTC onboard system which follows ERTMS/ETCS specifications and present patterns applied to our system.

1. 서 론

Design Pattern은 Software Design시에 일반적으로 많이 요구되는 재사용 가능한 일반화된 솔루션이라고 할 수 있다. 이 Design Pattern은 코드를 생성할 수 있는 최종 결과물이라기보다는 Design시에 발생하는 문제점들에 대해서 사용할 수 있는 해결 방법을 모은 설명이나 템플릿이라고 할 수 있다. 또한 개발자들에게 있어서는 공통의 표준화된 언어가 될 수 있으며 다른 시스템의 유사한 요구사항에 대해서 쉽게 적용할 수 있다. 본 논문은 ERTMS/ETCS Level 3의 시스템 요구 사양을 기반으로 현재 개발중인 CBTC 차상 시스템에 대하여 Harmony-SE process 중 Design단계에서 구조 및 기능별로 설계한 모델을 제시하며 이 과정에서 적용한 design pattern에 대하여 설명한다.

2. 본 문

2.1 CBTC 시스템

CBTC(Communication Based Train Control) 시스템은 궤도회로를 사용하지 않으며 지상과 차상간의 고용량 양방향 무선 통신을 지원하는 지상과 차상의 열차제어 시스템이다. 기존 궤도회로 방식은 열차가 궤도의 특정 부분만 점유하더라도 전체 궤도가 점유되는 것으로 처리되기 때문에 궤도의 사용이 비효율적이 된다. 또한 효율성을 높이기 위해서 궤도의 길이를 너무 작게 나누면 추가로 그만큼의 설비가 필요하므로 경제성이 떨어진다. 전송되는 정보의 양적인 면에서도 궤도회로를 이용한 방식은 그 양이 제한적이다. CBTC 시스템은 정밀하고 유연성있는 제어를 하도록 함으로써 궤도회로 방식의 한계를 극복하여 높은 정밀도로 열차를 검지할 수 있고 지상과 차상간에 연속적인 제어와 감시가 가능하여 최근에 기술 개발 및 적용되는 추세이다.

* LS산전(주) 중앙연구소

** LS산전(주) 교통SOC소사업부

2.2 Harmony Process와 Design Pattern

열차제어 시스템에서 사용되는 임베디드 소프트웨어는 알고리즘의 복잡성과 다른 일반 응용 프로그램과의 차별성을 요구하여 점차적으로 모델 기반의 개발 환경으로 옮겨가고 있다. 모델 기반의 개발 환경의 큰 장점으로는 시스템 개발 사이클의 시작 단계인 모델 단계에서의 알고리즘의 시험이 가능하여 구현 단계 이전에 모델의 정합성을 시험해 볼 수 있다는 점이다. 또한 이렇게 설계한 모델은 플랫폼에 독립적으로 개발되어 다양한 시스템 환경에 쉽게 변경하여 적용할 수 있다. 본 시스템에서 적용한 Harmony 프로세스는 모델 기반의 개발 방법론의 하나로써 Spiral 모델에 기반한 시스템 엔지니어링 프로세스이다. Harmony에서는 패러다임 독립적인 모델링 언어로서 SysML과 UML을 사용함으로써, 시스템 엔지니어링 단계로부터 소프트웨어 엔지니어링으로의 자연스러운 전이가 되도록 하며 시나리오 기반으로하여 개발과정 전체에 걸쳐 재사용된다.

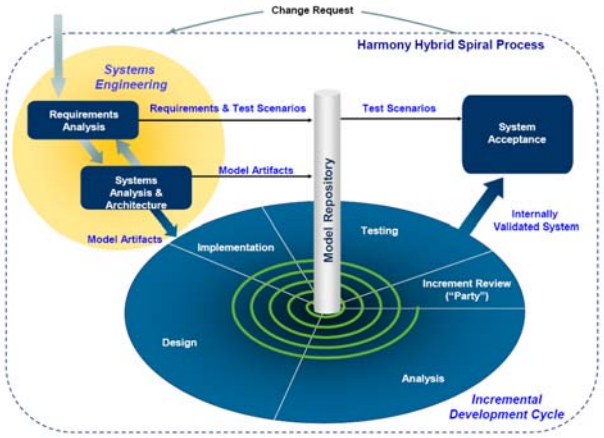

그림 1 Harmony Process Hybrid Spiral

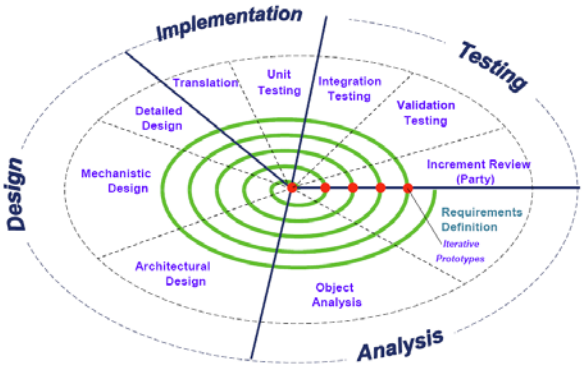

그림 2 Harmony Software Engineering

Design Pattern은 공통적으로 발생하는 문제에 대한 일반화된 솔루션으로써 여러 도메인에 걸쳐서 문제가 광범위하게 발생하여 충분히 일반화가 되어야 한다. 일반화 되지 못한 단일 도메인에 대한 것을 Analysis Pattern이라고 하며 이것은 주로 무엇을 패턴으로 하는가에 포인트를 둔 것임에 반해 Design Pattern은 얼마나 잘 그것의 요구사항을 만족시키도록 구현하는 최적화에 중점을 둔 것이라고 할 수 있다. 패턴은 주로 Harmony Process의 소프트웨어 엔지니어링 중에서 Design 단계에서 적용된다. 이 단계에서는 거시적 관점에서 미시적 관점으로 좁혀가면서 Architectural, Mechananistic, Detailed Design 각 단계별로 패턴을 적용하며 최적화 시키게 된다.

2.3 CBTC 차상 시스템 모델

개발중인 차상 시스템 모델은 System Engineering 단계에서 CBTC의 표준 문서인 IEEE 1474.1과 유럽 철도의 표준 사양인 ERTMS SRS 2.2.2를 바탕으로 시스템의 요구사항을 정의하였고 이를 바탕으로 시스템 외부의 Actor와의 상호 작용을 정의한 Use Case를 도출하였다. 그리고 시스템 기능 분석 단계에서 시스템이 수행해야하는 시스템 오퍼레이션을 표현하고 이를 확인 검증하는 단계를 수행했다. 그리고 구조 설계 단계에서 정의한 시스템 오퍼레이션을 서브 시스템에 할당하는 단계를 거쳐 구체화하였다.

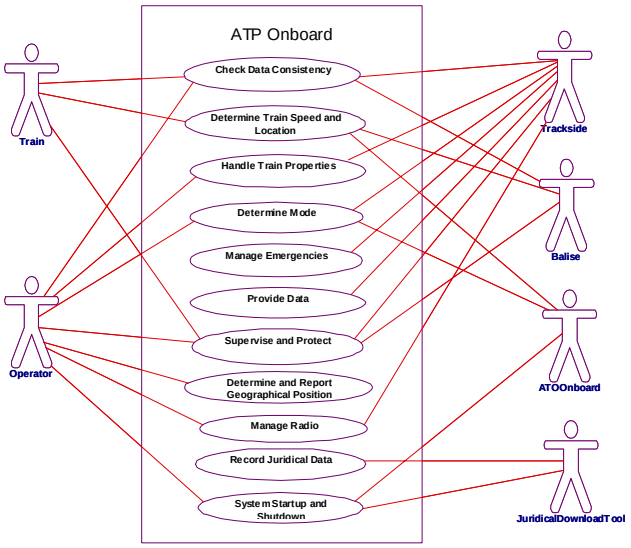

그림 3 시스템 Use Case

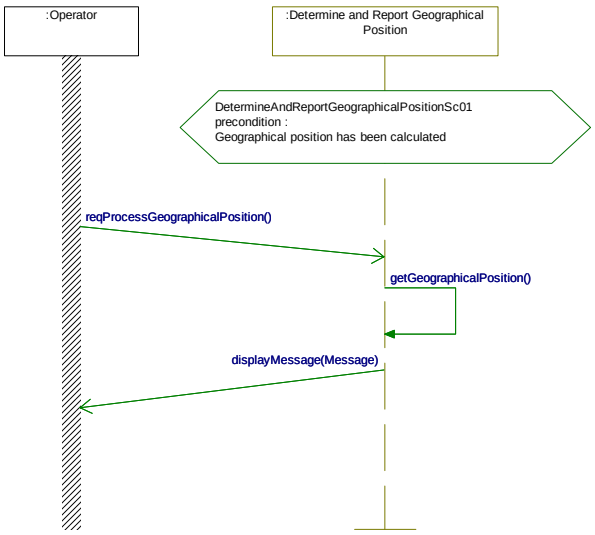

그림 4 시스템 오퍼레이션(지리적위치 보고)

이렇게 분석한 시스템 엔지니어링의 산출물을 가지고 나선형 개발 모델인 Software process 단계에서 시스템에 대하여 객체분석(Object Analysis)을 수행하기 위해서 Noun Strategy, Real-World Items, Transactions, Scenario등의 방법을 통해서 요구사항으로부터 시스템의 객체를 추출하고 클래스간의 관계를 정의하여 시스템 구조를 정의하였다.

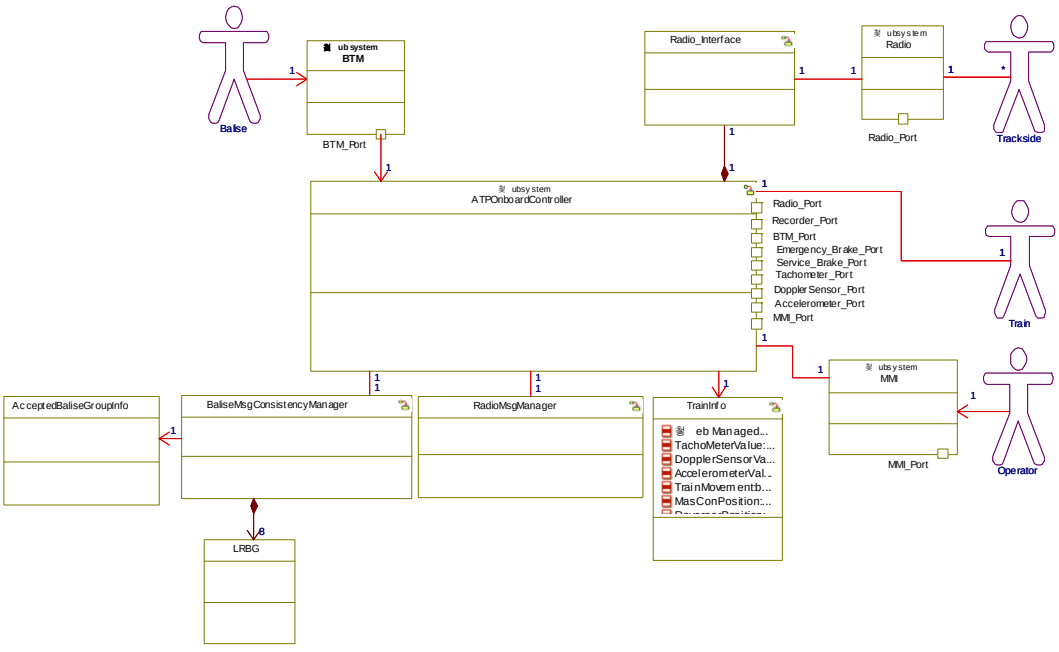

그림 5 Object Model Diagram

2.4 Architectural Design Patterns

Architectural Design은 Harmony Process중 Design 단계에서 첫 번째로 수행되는 구조 설계에 대한 것으로써 앞서서 구성한 모델을 기반으로 요구사항에 가장 잘 부합하도록 목적에 맞는 Architectural Design Pattern들을 적용하게 되었다.

2.4.1 Five Layer Architecture Pattern

그림6은 Five-Layer Architecture Pattern으로 전체 시스템의 구조를 정의한 그림이다. 이 Pattern은 임베디드 실시간 시스템의의 구조를 표현하는데 유용하기 때문에 사용되었다. 일반적으로 작거나 중간 정도 규모의 시스템에서 유사한 구조의 논리적 구조를 사용하여 시스템을 구성하면 개발자가 새로운 시스템의 구조를 파악하고 이해하는데 용이하다. 즉 차상시스템에서는 그림에서와 같이 시스템은 일종의 표준화시킨 다섯 개의 domain으로 표현하여 각 domain별로 시스템에 맞는 Layer를 구성하였다. 그리고 OnboardCommDomain 내부에 시스템에서 사용하는 무선 Radio와 차내의 통신을 위한 MVB, 그리고 직렬 IF를 위한 Serial를 정의하였다.

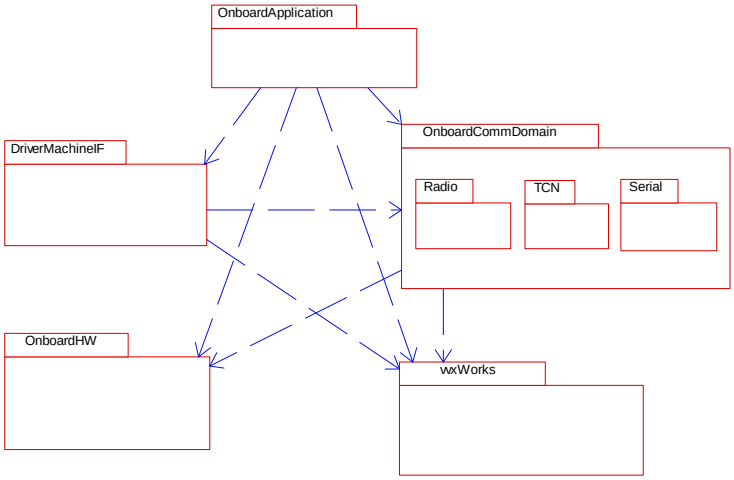

그림 6 Five Layer Architecture

2.4.2 Channel Architecture Pattern

Channel Architecture Pattern은 다음의 두 가지 경우에 대해서 적용을 하였다. 첫째는 채널이란 용어의 의미대로 일련의 데이터의 흐름이 단계별로 순차적으로 적용이 되는 경우에 부분적으로 적용이 되었다. 일반적으로 채널을 사용한 구성은 단순한 구조로 비교적 적은 비용으로 적용이 가능하고 고장 발생시에 대체가 용이하다. 그림과 같이 기본 구성을 무선 정보의 입력에 의해서 정보의 확인과 검증을 거쳐서 이동권한에 대한 MRSP(Most Restrictive Speed Profile)를 계산하여 Control Speed를 계산하는 채널을 구성하였다. 둘째로는 이 패턴을 확장하여 동일한 여분(Homogeneous Redundancy)을 갖는 구조적인 여분을 제공하도록 해 줌으로써 고 신뢰성과 안전성을 높이도록 변형하여 사용할 수 있다. 무선통신을 통해서 수신한 이동권한 정보를 이용하여 속도 제어를 수행하도록 채널은 구성하였으나 이에 대해 신뢰성과 가용성을 확보하기 위해서 그림 7과 같은 동일한 여분에 의한 구성의 적용을 검토중에 있다.

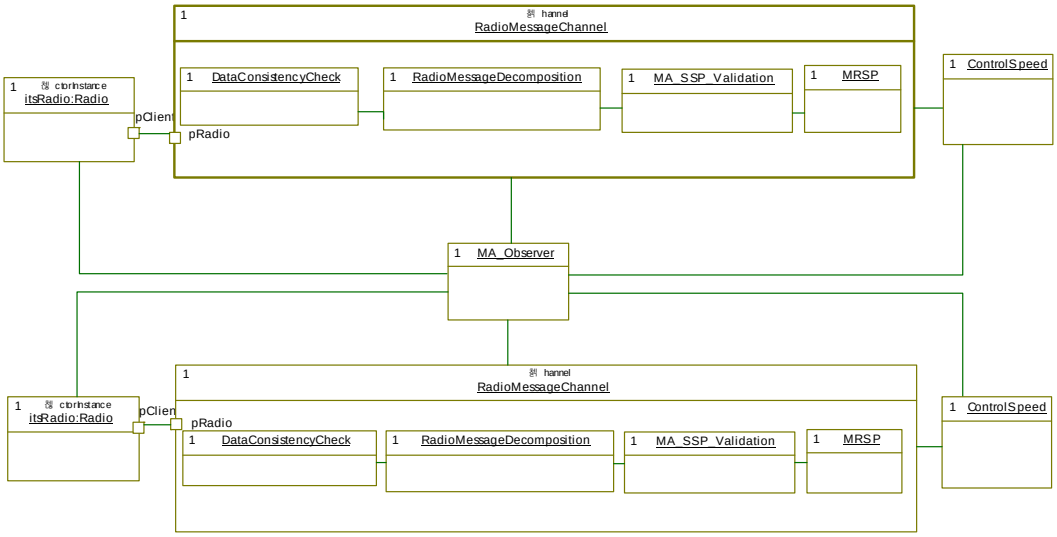

그림 7 제어 속도 연산

2.4.3 Monitor-Actuator Pattern

다음은 차상시스템의 속도 제어에 관한 부분으로 속도 센서로부터의 속도 정보 입력을 수신하여 검증 및 운행 조건과 비교하여 속도를 제어하도록 알람 또는 제동 출력을 하도록 감시하는 구조의 Monitor-Actuator Pattern에 관한 것이다. 여기서 Monitor부분은 속도 데이터의 허용 범위를 기준으로 감시하게 되며 고장을 검지하고 안전측 동작을 시킬 수 있도록 보완해 주는 역할을 하게 된다. 아울러서 속도값의 정합성 검증을 위해서 BIT(Built-in tests)를 포함하고 속도 제어의 부분에 제어 출력에 대한 출력 범위를 검사하는 클래스도 정의하였다.

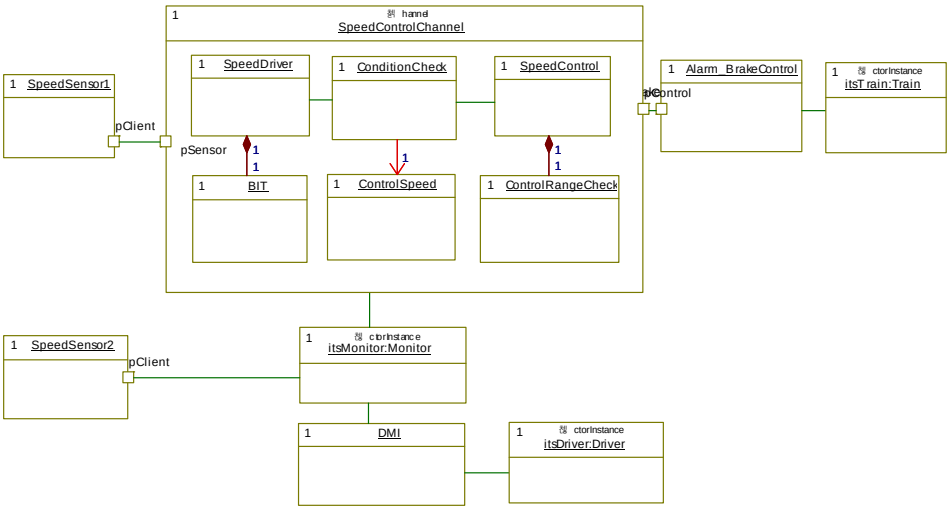

그림 8 속도 연산

2.5 Mechanistic Design Patterns

Mechanistic Design Patterns은 Design 단계의 두 번째 단계에서 실행하는 것으로 클래스와 객체 단위로 구성되어 use case와 같이 특정 요구 사항을 실행하도록 구성되어 있는 모델의 부분을 최적화하는 것이다. 차상 시스템에서는 그림 3와 같이 총 11개의 use case를 정의했으므로 각각의 use case 별로 Pattern을 적용하여 최적화하는 것에 대해서 고려하였다.

2.5.1 Adaptor Pattern

시스템을 구성하다보면 요구사항이 변경되었거나 기술적인 문제로 인터페이스가 변경되는 경우가 종종 발생하고 이때마다 해당 인터페이스 프로그램을 재 작성해야 하는 문제가 발생한다. 이러한 경우를 대비해서 서비스 사양을 실제 서비스를 구현하는 부분과 분리시키면 이러한 문제시 재작성의 수고를 덜어주고 향후 인터페이스 업그레이드 등에 유연하게 대처할 수 있다. 이때 사용하는 것이 Adaptor Pattern이다. 차상 시스템을 설계하면서 속도 센서의 경우에 열차의 사양 또는 정밀도에 따라서 다양한 속도 센서를 사용해야 하는 경우가 발생할 수 있다. 이에 대비해서 공통의 인터페이스와 adaptor를 구성하여 센서에 맞추어 서비스 요구를 변경하여 요구하고 센서로부터의 응답값을 변경하여 전달한다.

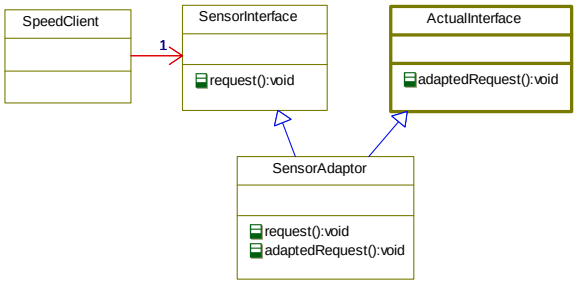

그림 9 속도 센서 인터페이스

2.6 Detailed Design Patterns

Design단계의 마지막 단계인 Detailed Design단계에서는 클래스 레벨에서 시스템의 최적화 작업을 한다. 즉 클래스 내부의 변수, 함수, 상태, Activity, 포트를 주로 타깃으로 하고 내부 데이터의 구조화, 내부 알고리즘의 구조화와 최적화, state machine의 최적화, 내부의 오류와 예외 처리등에 대한 부분을 중심으로 설계한다.

2.6.1 Any State Pattern

ERTMS기반의 신호체계 하에서 차상 시스템의 모드는 총 16개의 상태가 조건에 따라 전환하는 구조를 갖고 있어 state를 모두 표현할 경우에 대단히 복잡한 상태 천이의 구조를 보이게 된다. 이런 구조에서는 이해도 어려울뿐더러 개발 이후에 디버깅이 상당히 어렵게 된다. 여기서 적용한 Any State Pattern은 동일한 behavior를 표현하는데 직접적인 상태 변환을 묘사하는 것보다 상태 전이 구조를 단순화시켜 줄 수 있어서 적용하였다. 그림10은 이 Pattern을 적용하여 모드 전환을 나타낸 그림이다.

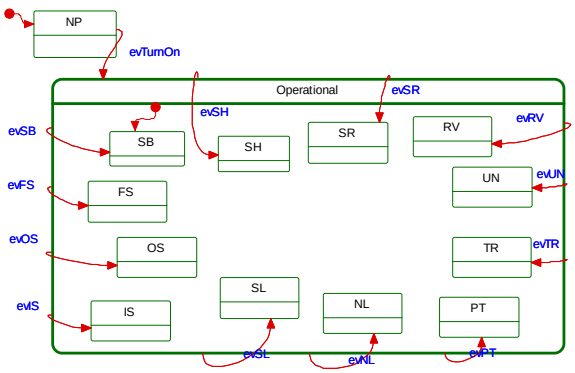

그림 10 Mode 전환

2.6.2 Polling State Pattern

데이터를 취득하는 등의 주기적인 일을 처리하는데 유용한 것이 Polling State Pattern이다. 이 Pattern은 단독으로 쓰이기보다는 동시에 다른 제어나 데이터 처리를 수행하면서 주기적으로 데이터를 취득하던지 전송하는 등의 일을 병행할 때 사용이 된다. 차상 시스템의 모델링에서도 입력 데이터의 취득이나 제어 출력에 사용을 하며 아래에 첨부한 예는 주기적인 위치보고를 하도록 설정이 되어있을 때 2초 주기로 위치보고를 무선으로 전송하도록 설계한 그림이다.

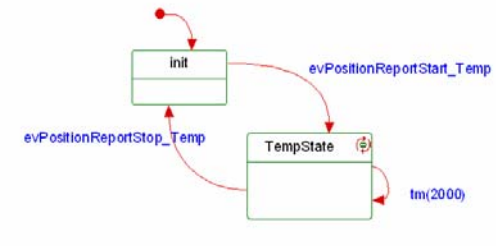

그림 11 주기적 위치보고

3. 결론

본 논문에서는 CBTC 차상시스템을 Harmony Process에 따라서 개발하면서 Analysis단계의 산출물을 이용하여 Design 단계를 수행하면서 세 단계(Architectural, Mechanistic, Detailed Design)에서 각각 최적화를 수행하기위해서 Design Pattern을 적용하는 방법을 보였다. 물론 패턴 자체의 적용이 없이도 목적하는 기능을 구현하는데는 어려움이 없을 수 있으며 때로는 인지하지 못한채 사용을 하는 경우도 있다. 다만 설계 단계에서 미리 필요한 패턴에 대한 충분한 지식과 경험을 보유하면 디자인에 소요되는 시간을 절약할 수 있으며 이미 검증된 설계를 반영할 수 있어 전체 개발에 소요되는 자원과 노력을 줄일 수 있다. 또한 패턴을 적용한 시스템은 타 개발자가 쉽게 이해가 가능하여 시스템 보완이나 추가 업그레이드도 용이하다. 본 논문에서 적용한 패턴들외에도 범용화된 많은 유용한 패턴들이 있으며 향후 개발 사이클이 반복되어 진행되면서 필요에 따라서 추가적인 패턴들의 적용을 검토할 것이다.

[참고 문헌]

1. Bruce P. Douglass, "Real-Time UML Workshop for Embedded Systems",ELSEVIER
2. Bruce P. Douglass, "Real-Time Design Patterns",Addision Wesley
3. "IEEE Standard for CBTC Performance and Functional Requirements", (IEEE Std 1474.1-2004)
4. UIC(2002), "ERTMS/ETCS System Requirements Specification 2.2.2", ERTMS User Group
5. Hans-Peter Hoffmann(2006), “HARMONY-SE / SysML Deskbook”, Telelogic
6. 양찬석(2007), “ERTMS/ETCS 기반의 CBTC 시스템 기본설계”, 한국철도학회 추계 학술대회지
7. Ali Ebnenasir(2007), "Pattern-Based Modeling and Analysis of Failsafe Fault-Tolerant in UML", IEEE High Assurance Systems Engineering Symposium